

Amendments to the Specification

Please replace the paragraph that begins on Page 1, line 9 and carries over to Page 2, line 9 with the following marked-up replacement paragraph:

-- The distribution of a single program, so that a portion of the program executes on more than one computer, has become more pervasive as "desktop" computers have become more powerful. Figure 1 demonstrates an example of a computer network containing several distributed computers upon which an application can execute. While most computer networks are many orders of magnitude larger, this small network is used as an example. Presently, many computer systems allow objects to communicate over a network. One method of distributing object oriented programs across computer networks is Automatic Object Distribution (AOD), described in the commonly assigned, copending patent application having serial number 08/852,263, now U. S. Patent 6,157,960, entitled "Technique for Programmatically Creating Distributed Object Programs" "A Process for Running Objects Remotely", filed on May 6, 1997. -

Please replace the paragraph that begins on Page 11, line 16 and carries over to Page 12, line 6 with the following marked-up replacement paragraph:

-- Figure 5B represents the computer system of figure 5A after it has been distributed using the AOD process described above. Objects X 502 and Z 504 remain in computer 501, but object Y 503 has been moved to another computer 509. The AOD process knows from examining the objects' byte codes that object X 502 will be calling object Y 503, so it creates two proxies for object Y, one on computer 501 referred to as Y' 510 and one on computer 509

referred to as Y" 511. Because the AOD process also knows that object Z 504 will be passed as a parameter on the call from X 502 to Y 503, the proxies for object Y are enhanced by the present invention to create the appropriate proxies for object Z 504 so that object Z may be passed as a parameter on the remote call to object Y without needing to need to lock and serialize object Z. Additionally, since the logic is added by the AOD process to the proxies created for object Y, no programmer intervention is necessary to accomplish this efficient remote call using object Z, allowing the entire program to be written as if all its objects are to reside on one computer. --

Please replace the paragraph that begins on Page 14, line 15 and carries over to Page 15, line 2 with the following marked-up replacement paragraph:

-- In figure 4B, X 403 is shown calling a method of Y 404 to read a stream of data from an input source such as a diskette. (The input source may be any source of data that may be represented by a data stream object.) The call 407 is actually made to proxy object Y' 405, which passes the call via Remote Method Invocation (RMI) or some other standard remote calling method 408 to Y" 406. Y" then translates the call into the semantics that the actual object, Y 404, requires and then passes 409 the call to it. Y returns the result 410 in the form of a reference to a data stream object 418, which is accessed to obtain the data in the steam stream. --

Please replace the paragraph on Page 15, lines 11 - 20 with the following marked-up replacement paragraph:

-- Upon creation 421 of the network connection 414, proxy object Y' 405 understands that the connection is for data that is to be returned via a data stream object, and builds the data

stream object 422 such that when it is accessed to obtain data 423, the data 424 is read 415 from the socket connection. (In the preferred Java™ embodiment, a system call 416 may be used to build the data stream object, however other methods may be used in the preferred or other embodiments.) Y' then returns 417 this locally-created data stream object 422 to X 403. Thus X receives the requested data stream object and accesses it as if its source were local to machine 401, even though the data is actually local to machine 402 and being provided via a network connection 414. —

Please replace the paragraph on Page 19, lines 5 - 11 with the following marked-up replacement paragraph:

-- Upon receiving the remote call from Y' 510, Y" 511 creates a proxy Z' 512 for object Z 504 in machine 509, and creates 515 a reference table entry in which the key Z' returns a remote call reference to the proxy Z" 513 which was created by Y' 510. Then when object Y" 511 translates the call into the semantics of object Y 503 and invokes object Y, a reference to proxy object Z' 512 is passed 516 as the parameter. Thus object Y will invoke object Z' 512 when object Y's invoked method invokes the object passed in as a parameter. —

Please replace the paragraph that begins on Page 19, line 19 and carries over to Page 20, line 3 with the following marked-up replacement paragraph:

-- When object Z" 518 is 513 is invoked 518, it uses the reference table entry which was created earlier 524 by object Y' to determine where the call is to be directed. By looking itself up in the table 519, object Z" receives a reference to object Z 504. Using the received reference,

object Z" translates the received call into the semantics of object Z and invokes 520 object Z. Thus, object Y 503 has successfully invoked object Z in the course of its invocation from object X 502, even though object Y resides on a different computer than objects X and Z. --

Please replace the paragraph on Page 20, lines 4 - 12 with the following marked-up replacement paragraph:

-- Object Z 504 returns the result 521, if any, of the invocation to object Z" 513, which returns said result 522 to object Z' 512, which returns 507 said result to object Y 503. Then when object Y finishes the method which was invoked from object X 502, it returns the result 523, if any, of said invocation to object Y" 511, which returns said result to object Y' 510, which returns 508 said result to object X 502, thus this completing the object X's method invocation to object Y 503, which also updated object Z 504. The proxies do all the work so that the original objects X, Y, and Z may be programmed as if they all reside on one computer. --